TrueClick: Automatically Distinguishing Trick Banners from Genuine Download Links

Sevtap Duman
Northeastern University
Boston, MA
sevtap@ccs.neu.edu

Kaan Onarlioglu
Northeastern University
Boston, MA
onarliog@ccs.neu.edu

Ali Osman Ulusoy
Brown University
Providence, RI
ali_ulusoy@alumni.brown.edu

William Robertson
Northeastern University
Boston, MA
wkr@ccs.neu.edu

Engin Kirda
Northeastern University
Boston, MA
ek@ccs.neu.edu

Abstract

The ubiquity of Internet advertising has made it a popular target for attackers. One well-known instance of these attacks is the widespread use of *trick banners* that use social engineering techniques to lure victims into clicking on deceptive fake links, potentially leading to a malicious domain or malware. A recent and pervasive trend by attackers is to imitate the "download" or "play" buttons in popular file sharing sites (e.g., one-click hosters, video-streaming sites, bittorrent sites) in an attempt to trick users into clicking on these fake banners instead of the genuine link.

In this paper, we explore the problem of automatically assisting Internet users in detecting malicious trick banners and helping them identify the correct link. We present a set of features to characterize trick banners based on their visual properties such as image size, color, placement on the enclosing webpage, whether they contain animation effects, and whether they consistently appear with the same visual properties on consecutive loads of the same webpage. We have implemented a tool called TrueClick, which uses image processing and machine learning techniques to build a classifier based on these features to automatically detect the trick banners on a webpage. Our approach automatically classifies trick banners, and requires no manual effort to compile blacklists as current approaches do. Our experiments show that TrueClick results in a 3.55 factor improvement in correct link selection in the absence of other ad blocking software, and that it can detect trick banners missed by a popular ad detection tool, Adblock Plus.

1. INTRODUCTION

Internet advertising, both in traditional forms such as web banners and email campaigns, and more recent social media and mobile marketing schemes, is a rapidly growing, lucrative business [27]. Publishers and application developers find it increasingly easy to integrate advertising into their content and, consequently, attackers have leveraged this channel as an efficient mechanism for distributing malware. Naturally, the computer security community has shown an increasing interest in this field as well. Researchers have produced a large body of work to address critical security issues surrounding Internet advertising, such as sandboxing of advertisements from the actual content, resolving user privacy concerns, and preventing ad fraud.

While the security community has primarily focused on finding solutions to the technical side of the problems around Internet advertising so far, the *human factor* in Internet advertising and the class of attacks that attempt to exploit an Internet user's perception have largely been left unexplored. One well-known instance of such an attack is the widespread use of *trick banners* [32]. Trick banners are advertisement banners that are crafted to deceive and mislead users into clicking on them, potentially linking to a malicious domain or a malware executable. While trick banners have traditionally come in the form of colorful and animated messages, or as pop-ups imitating application messages, a more recent and pervasive trend is to imitate the "download" or "play" buttons in popular file sharing sites (e.g., one-click hosters, video-streaming sites, bittorrent sites) in an attempt to trick users into clicking on these fake banners instead of the genuine download link.

Previous work has explored user behavior and security awareness when browsing the Internet, and shown that trick banners found in file sharing sites are effective at tricking even technically sophisticated users who had previous familiarity with file sharing sites used in the study [37]. This study concluded that trick banners pose a significant security risk to ordinary Internet users, and even to those with a heightened security awareness. Indeed, abuse of advertisement banners in this way (a practice that is also referred to as "malvertising") has been recognized as a current and effective attack vector [42, 45]. It has also been shown that rather than using complex exploitation techniques, simply buying ad space is an easy and effective way for attackers to spread malware and quickly victimize a large number of Internet users [18]. Numerous attacks on high-profile websites and ad networks in recent years demonstrate that the problem is real and is actively being exploited [36, 50].

In this paper, we explore the problem of automatically

ACSAC '14, December 08 – 12 2014, New Orleans, LA, USA

http://dx.doi.org/10.1145/2664243.2664279

assisting Internet users in successfully detecting malicious trick banners, focusing on distinguishing fake download buttons ubiquitously found in popular file sharing websites from genuine download links. We first identify a set of features to characterize trick banners based on their visual properties such as banner size, color, placement on the webpage, whether they contain animation effects, and whether they consistently appear with the same visual properties on multiple copies of the same webpage obtained with separate requests. We then leverage these features in an approach combining image processing and machine learning to automatically detect trick banners on a webpage.

We implement our system in a prototype Firefox browser extension called TrueClick, evaluate its effectiveness on a data set of 259 banners collected from 88 file sharing websites, and demonstrate that TrueClick achieves a 96.97% true positive rate given a false positive rate of 3.03%. Note that unlike state-of-the-art ad blocking, our approach does not require a priori blacklisting of advertising domains, or any other manual classification of known banners. After an initial training phase, it operates in a completely automated manner by analyzing the visual properties of banners. In other words, the approach we propose is complementary to existing blacklisting approaches, and can support them in identifying previously unknown trick banners. Moreover, TrueClick does not rely on examination of the source code of webpages or the structure of the DOM tree. Instead, it utilizes image processing techniques to capture and analyze webpages *as the user sees them*, and therefore is not affected by attempts to thwart detection through dynamic modifications to the page.

In summary, we make the following contributions in this work.

- We present five visual features that can be used to characterize trick banners and experimentally demonstrate that these features can be used in practice to distinguish trick banners from genuine download links.
- We present a novel methodology for automatically and reliably distinguishing trick banners from genuine download links by using a combination of image processing and machine learning.
- We describe a prototype implementation of our solution as a Firefox browser extension called TrueClick that can effectively guide users toward finding and clicking on the genuine download link in a file sharing website littered with trick banners.
- We evaluate the usability of TrueClick with a user-study, and demonstrate a 3.55 factor improvement in selection of benign links in the presence of trick banners. We also show that TrueClick is able to detect trick banners missed by a popular ad detection tool, Adblock Plus.

2. PROBLEM STATEMENT

A trick banner is generally defined as any Internet advertising banner with a deceptive visual appearance, crafted to lure users into clicking on them [32]. They often do not contain any indication of the identity of the advertiser or the advertised service or product. Trick banners are known to integrate well with the look and feel of the website they appear on, and often imitate popular applications, operating system windows, and pop-up messages.

In this work, we focus on a specific pervasive class of trick banners: fake download buttons found on file sharing websites, which have recently been shown to be effective at tricking even users with security expertise [37]. Figure 1 shows various examples obtained from popular file sharing services. Note that the techniques we present in this paper are sufficiently generic to be applied to other kinds of trick banners as well. We believe that fake download buttons represent an up-to-date manifestation of the more general trick banner problem, and pose a challenging research task because of their tight integration with the file sharing sites they are displayed on. To illustrate, one of the sample trick banners taken from The Pirate Bay (Figure 1, lower left corner) displays identical replicas of the correct download links (i.e., the links "GET THIS TORRENT" and "ANONYMOUS DOWNLOAD"), making it an especially difficult trick banner to spot for an unsuspecting user.

Also note that banner design (and user interface design in general) for attracting users and maximizing click-through rates has been extensively studied in computer science and other, non-technical fields [5, 8, 10]. Our use of the term *trick banner* in this paper could refer to images with either benign or malicious intent. In other words, trick banners we examine could be crafted with malicious intent, such as directing a user to an attack site or downloading malware to her computer. Or, they could simply be used as a device for artificially inflating the number of visitors to a destination website, without explicit malicious intent. In this work, we do not aim to verify whether the trick banners lead to malicious destinations (other recent work has explored this aspect of the problem [30]), nor do we visit the sites or download the files they link to. Instead, our goal is to detect trick banners regardless of their purpose, and distinguish them from legitimate, genuine download links.

We divide the threat model we consider for this work into two scenarios. In the first scenario, an Internet user visits a file sharing website with the intention to download a specific file. We do not make any assumptions about the security expertise or awareness of the user. The pages on the website can contain any number of trick banners and regular advertisements, and one or more correct download buttons defined as the link to the content the user intends and expects to download. In cases where the website requires the user to step through a number of different pages to complete the download, we take the links that lead the user closer toward the final download link as the correct one.

The second scenario is identical to the first except that the website in question does not actually contain any correct download links. Such sites could be found on the Internet, for example, as part of a well-known scam scheme in which a scammer creates webpages that contain detailed descriptions of various content despite not including any actual download links, possibly in an attempt to trick search engines and steal clicks from users for ad revenue.

In both of these cases, regardless of whether a correct link exists or not, our system analyzes the webpage and determines the page regions that contain trick banners. If trick banners are detected, they could be marked with warning cues to alert the user or blocked entirely, depending on implementation choices or user preferences.

Finally, we would like to point out that there exist other types of attacks involving malicious modifications to websites and browser user interfaces that aim to make users in-

Figure 1: Four trick banner examples taken from popular file sharing sites The Pirate Bay, Rapidgator, and FilesTube that imitate the look and feel of genuine download buttons to deceive users.

advertently click on incorrect or malicious links. For example, various forms of clickjacking attacks compromise the visual and temporal integrity of the pointer cursor or browser's display to this end [24]. The problem we aim to address in this work is separate from those attacks in that, in our threat model, the system's integrity is not compromised; instead, trick banners exploit weaknesses in human perception to trick users. In clickjacking attacks, the user's system is technically crippled, making her unable to click on the correct link even if she can identify it. However, in the attacks we aim to address, the user *willingly* clicks on a trick banner, thinking that it is a genuine download link. While defenses against both types of attacks are necessary for secure browsing, we only explore the latter problem in this work.

3. BACKGROUND & RELATED WORK

Before we explain our approach to trick banner detection, we present related work and the state-of-the-art in this field in order to highlight the differences between our work and current advertisement detection systems.

Internet Advertising and Trick Banners. There is a large body of prior work on Internet advertising and the technologies around it, both in computer science and other non-technical fields. The effectiveness of various types of ad banners and ways to influence user click-through behavior have been studied widely in marketing, finance, psychology, and related fields [8, 12]. In a recent work, Onarlioglu et al. [37] presented a study that investigates the computer security implications of trick banners, and showed that trick banners can mislead even technically sophisticated Internet users and expose them to attacks.

Advertisement Blocking and Filtering. A simple method of blocking trick banners, and advertisements in general, is to disable prerequisites for displaying such content in web browsers. This can involve disabling image loading, blocking Flash Player and similar browser plugins, or blocking JavaScript code used by advertising networks through widely available browser security extensions [25]. Similarly, Web proxies could be deployed to filter out trick banners before they could be displayed in the browser [3, 4]. While these solutions are effective at blocking trick banners, they also significantly impair the user's browsing experience, or even render many websites nonfunctional, as the World Wide Web today makes extensive use of multimedia and dynamic content. Moreover, deploying HTTP proxies might not be accessible to the average Internet user, or might not be a viable option on more restricted mobile devices.

An alternative approach to the problem is using specialized ad filtering software that is often included as part of commercial antivirus suites, or designed as open-source web browser extensions such as the popular Adblock Plus [1]. On the one hand, these solutions offer the ability to selectively block offending content and therefore provide improved usability over the previously discussed solutions. On the other hand, they typically detect links to advertisements by consulting various blacklists and whitelists that must be continuously maintained and updated, which often involves significant manual labor. TrueClick instead performs trick banner detection based on the visual characteristics of such content in a completely automated manner and, thus, is able to detect trick banners that have not previously been classified into blacklists or whitelists.

Security Uses of Visual Features. Another body of work applies image comparison techniques and visual similarity metrics to different Internet security problems. For instance, various groups have investigated phishing site detection techniques based on visually comparing suspected phishing pages to their legitimate counterparts [9, 33, 34, 49], and Gargiulo and Sansone [13] use image processing to extract visual and text features from spam emails. In contrast, we identify visual features specifically tailored to detect trick banners and implement them in TrueClick.

Doppelganger [43] explores the results of different cookie policies on websites by transparently mirroring the user's web session to create two conceptual browser windows, one with cookies enabled and one without. They then compare the two to investigate the impact of accepting cookies from a given website on the content displayed. Likewise, TrueClick utilizes a detection feature based on comparing multiple views of a webpage, as we explain in Section 5.1.

Securely Isolating Ads and Applications. A large number of studies aim to protect sensitive web content from potentially malicious third-party ads by sandboxing the ads displayed on the page. Several projects use language containment and static policy enforcement to restrict JavaScript features used by ad networks [2, 11, 19, 23, 31], while others perform dynamic policy enforcement [15, 35, 46]. Likewise, researchers have investigated the ad ecosystem on mobile devices and proposed approaches that aim to isolate third-party ad libraries from mobile applications [16, 29, 38, 44]. Other approaches to create general secure mashups of advertisements and applications include secure browser architectures and browser-based operating systems [17, 39, 48].

In another set of studies, researchers explore the privacy issues around Internet advertising, and propose techniques to deliver targeted ads while limiting the exposure of privacy-sensitive user information to ad networks [20, 21, 28, 40, 47].

These studies have the common goal of protecting applica-

Figure 2: The image extraction process illustrated on a real trick banner. Step 1 detects four superfluous sub-regions on the banner, Step 2 then corrects this error, and matches the fragments to the actual image.

tions against uncontrolled and potentially malicious ad code. While the isolation they provide is clearly essential for secure deployment of online ads, the measures they employ do not provide protection in a scenario in which Internet users are deceived and misled to click on a trick banner that links to a malicious destination. In contrast, TrueClick addresses this problem by analyzing the ad banners on a web page and guiding users to identify the genuine download link among a set of trick banners.

4. IMAGE EXTRACTION

Before we can compute any features on potential trick banner images, we first need to identify and extract all of the image regions on a webpage, which are subsequently fed to our trick banner classification system. Note that simply searching for HTML image tags in the page source is not sufficient to perform this task correctly, because some of the banners may be loaded dynamically by JavaScript, or they may come in non-image formats like Flash files or regular links stylized to look like buttons. In this section, we briefly explain the details of this process.

The image extraction technique we propose in this work is a two-step process. Initially, we leverage well-known image processing techniques designed for this task, which follow the common pipeline of edge detection, region filling and connected component analysis [14], and banner region identification on the webpage. However, the enormous variability in webpage content often does not allow a single generic image processing pipeline to perform perfectly in all cases, and our early experiments indicate that extracting image regions solely through image processing usually falls short. For example, when faced with a trick banner that displays several button lookalikes in a single image file, the aforementioned image processing pipeline yields multiple detection results for each disconnected component in the image. (See the banner presented in Figure 2 for a real example.) Similarly, more sophisticated visual designs could result in a large number of superfluous detections of small sub-regions on a single image (or, conversely, missed portions of banners), which would later result in unnecessary detection feature computations or inaccurate results. Although image processing methods that are customized for each specific website can be devised to improve extraction accuracy and performance, such an approach would be time and effort-intensive, without any clear indication as to how the extraction scheme could be adapted to the future banner designs.

Instead, based on the observation that while the discussed image processing techniques are imperfect, they rarely completely miss entire banner regions, we employ a second step to correct for partially detected and fragmented banners. This involves collecting and temporarily caching all of the actual image files requested from the web server for a given webpage to form a small image database. Next, the possibly fragmented banners extracted using the initial image processing step are matched to the images in the database. This matching is performed using the SURF (Speeded-Up Robust Features) feature detector and descriptor, which is widely used for object detection and recognition in the computer vision community [6]. Our experiments show that even when a significant portion of the banner is fragmented, SURF is able to match the banner to the correct image in the database. When a match is found, or in other words the extracted region is detected as a fragment of a larger image, the extracted banner image is simply replaced with the correct image from the database.

This process is illustrated in Figure 2. In Step 1, the image processing pipeline incorrectly identifies four separate regions in the banner. Later, in Step 2, these four extracted regions are all matched to the actual banner image stored in the database (and are replaced by it), resulting in an accurate banner extraction.

We must point out that even in the presence of the image database we build in Step 2 of this process, the banners extracted through image processing in Step 1 still provide valuable information for detecting trick banners; specifically, those that come in non-image formats. For example, in our experiments we observed static banners delivered in Flash files, or as regular links in HTML iframes stylized to look like buttons. While image processing can identify and extract such non-traditional banners, attempts to match them to image files in Step 2 would fail since there does not exist a corresponding image file on the webpage. Therefore, in cases where we cannot find any successful match in Step 2, we do not discard the regions extracted in Step 1 but instead input them to the classifier as-is.

5. TRICK BANNER CLASSIFICATION

Once possible trick banner regions have been identified on the webpage, five visual features are extracted to help classify each region as either a trick banner or a genuine download link. These features include *a*) image color, *b*) image size, *c*) image placement, *d*) presence of animation, and *e*) image differences between consecutive page loads. In the remainder of this section, we provide details on each of these features, explain why they are useful for distinguishing trick banners from genuine download links, and then present the classification approach we adopt in this work.

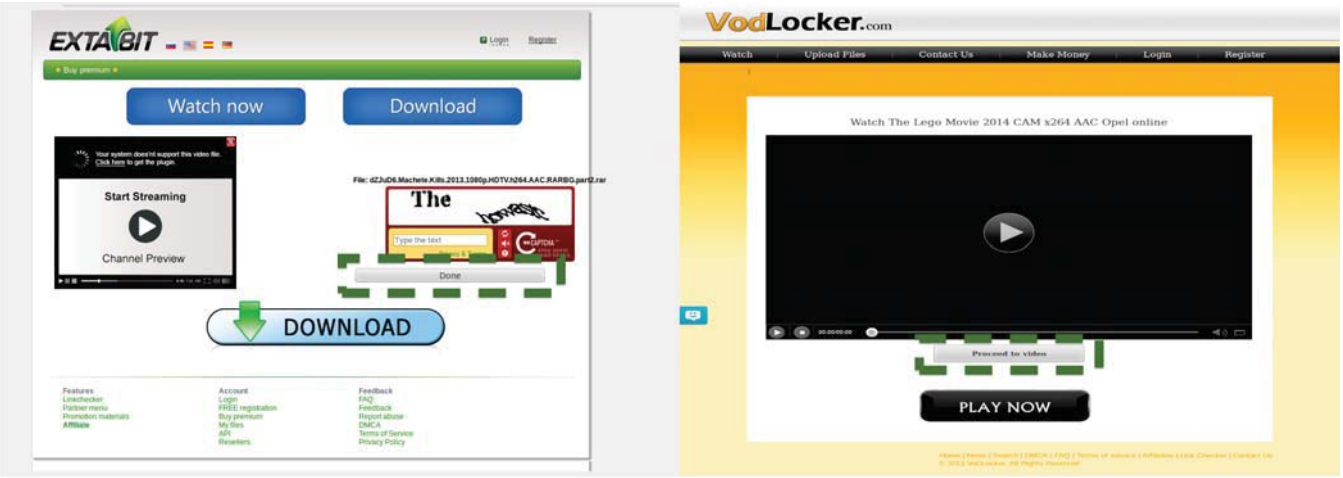

Figure 3: Sample webpages that illustrate that EMD scores vary with the color theme of the websites, and should only be compared after normalization. The correct banners are marked with dashed boxes.

5.1 Features

Color. Trick banners are often not designed by the site owners, and are usually served by third-party advertising networks just like regular ad banners. Consequently, the designers of trick banners do not know the exact website their banners are going to be displayed on, which leads them to follow common webpage theme specifications in their visual designs. As a result, trick banners often do not fit the general color theme of the website, but instead display distinctive color signatures. In contrast, genuine download buttons usually cohere to the overall website colors. This distinction suggests that banners can be classified based on their color similarity to that of the overall website.

This classification requires first a description of the colors inside the banner region. Experiments indicate both genuine and trick banners can be quite complex in terms of the color patterns they contain. Most banners are composed of a multitude of colors, highlights, and gradients, and also include other small images. Color histograms are ideal for the purposes of capturing the global color features of the banner. Histograms are constructed simply by binning the color of each pixel in the banner region. The histograms are finally normalized by their total mass such that they are invariant of the number of pixels in the banner region.

Classification based on color histograms also requires a method to compute the similarity between histogram pairs. In our work, we compute color histogram similarity using the Earth Mover's Distance (EMD) [41], which is widely used in content-based image retrieval applications. In short, EMD is a metric between two distributions (of equal total mass) that measures the minimal cost incurred to transform one distribution to the other. The comparison is made between each banner and the whole page histograms. We made our computations on the RGB channel.

Note that the EMD score of a banner is computed with respect to the color histogram of the overall webpage and, thus, this score provides a measure *relative* to the other banners within the webpage. For instance, on the left-side webpage in Figure 3, the EMD score of the correct banner (marked with dashed lines) is 17443.67, and the trick banners on the same webpage have higher scores than the correct banner. However, we cannot generalize this outcome and consider scores higher than 17443.67 obtained from different pages to indicate a trick banner. To illustrate, the webpage on the right side of Figure 3 has a correct banner that has an EMD score of 19773.83, and while this value is still lower than the EMD scores of the trick banners on the same webpage, it is higher than 17443.67. Therefore, a normalization is required so that the EMD scores of all banners in training and classification are comparable. Mapping the EMD scores of banners in each website to a fixed interval such as $[0, 1]$ is sufficient for this purpose.

Size. Websites often reserve fixed sections in their visual layouts where advertisements can be displayed. The sizes of these reserved spaces are not strictly controlled. However, in accordance with the Interactive Advertising Bureau online advertisement size guidelines [26], these sections are usually large, horizontal, or vertical rectangular regions in which standard web banners that advertising networks serve can fit. In contrast, genuine links on a file sharing website usually take the form of a single, relatively small image that serves as a download button. Consequently, in order not to throw off their disguise by displaying unusually large fake buttons that contrast with the rest of the page's style, many trick banners resort to tricks such as using large empty borders around a smaller fake button image, or including two or more button images on a single banner as if they were separate clickable entities. Thus, image size is a strong feature for distinguishing trick banners from genuine buttons.

We measure the size of these images in the x and y dimensions in the number of pixels. In order to deal with varying webpage sizes, we first normalize the numbers to $[0, 1]$ with respect to the absolute size of the enclosing webpage.

Placement. Navigation links on a website, including the genuine download buttons, are tightly integrated with the rest of the site's content. In contrast, advertisement banners are often laid out separately in reserved spaces in order not to interfere with the coherence of the website's interface and content. They are usually placed at the page header, footer, sidebars, or are otherwise isolated from the actual page content. Therefore, the position of the banner can be used as an indicator for trick banners. We use the x and y

positions of the geometric center of the banner as the placement feature. Similarly to the size feature, we first normalize the values to $[0, 1]$.

Animation. Another significant indicator for trick banners is the use of animations that are employed to draw the attention of the user. Animated banners rapidly display a sequence of images, typically in the form of GIF images. Most genuine download links do not contain animations. In fact, during our study we didn't encounter any genuine links that contained animations.

Animation is a binary feature, indicating whether or not the banner is animated. The presence of animation is detected by first checking whether the banner image format allows animations. If it does, the number of frames embedded in the file is used to decide the presence of animation. This animation check is performed only on the images that were selected from the database because it is not possible to reach fragmented image information unless it matches with a cached image.

Visual Differences in Multiple Page Views. A common method for deploying advertisements on a page for a website owner is to utilize advertising networks, which serve a different ad banner every time a user visits the page. Similarly, large content publishers may use their own advertising infrastructures that rotate the banners displayed on the page each time the page is visited. Consequently, the visual contents of banner spaces on webpages tend to be very dynamic, often changing every time the page is loaded or refreshed in the browser. In contrast, the user interfaces of webpages are often comprised of a fixed set of images that seldom change once the website's design has been finalized. To promote usability and provide a consistent user experience, menus, navigation links, and buttons on the page are placed at specific positions and use static images.

We take advantage of the dynamic characteristics of banners and the static nature of the rest of the UI elements on a webpage to propose a trick banner detection feature based on comparing multiple views of a single webpage. Specifically, we first take two screen captures of the same page obtained through two separate requests to the web server. We then visually compare them, extract the parts that have changed between the requests, and mark those as potential banner regions.

5.2 Classifier

A binary classifier is a function that takes as input a set of features, such as the visual features described above, and outputs a binary decision – in this case, *trick banner* or *non-trick banner*. The machine learning literature offers a wide variety of binary classifiers [22]. In this paper, we choose to use the popular random forest classifier, and train it using the method proposed by Breiman [7]. The details of our training data collection methodology is explained in Section 7. Note that although the random forest is used for all experiments in this paper, we have observed that the results are comparable using other state-of-the-art classifiers such as the support vector machine.

6. IMPLEMENTATION

We implemented the trick banner detection methodology we have discussed above in a prototype system called TrueClick, as a Firefox browser extension that uses external image processing libraries. In this section, we explain the implementation-specific details of our system.

6.1 Overview

TrueClick is implemented as a browser extension that runs on demand when the user visits a file sharing website containing trick banners and clicks on a button to activate the system. Once the analysis of the banner images is complete, TrueClick can either mark the detected trick banners as such, or block them entirely. In this prototype implementation, we elected to visually obscure the trick banners from the user.

An overview of the architecture of TrueClick is presented in Figure 4. When the user triggers the analysis on a given web page, the *Screen Grabber* and *Image Grabber* components first take screenshots of the webpage and cache all image files downloaded from the web server in an image database, respectively. Then, the screenshots and image database are input to the *Image Extractor* component, which identifies the banner regions on the screenshots, and attempts to match them to the files in the database, as previously explained in Section 4. Once all banner regions are detected, they are sent to the *Feature Extractor* which computes the five trick banner detection features on each image. Finally, TrueClick runs the resulting feature vectors through its classifier and determines the regions on the webpage where trick banners are displayed.

Note that during this process, all extracted images smaller than 16×16 (i.e., the standard size for favicon files) are discarded in order to speed up the detection process and display more relevant warnings to the user since they are most likely not banners.

In the following, we elaborate on the details of the Screen Grabber and Image Grabber components.

6.2 Screen Grabber

The Screen Grabber component is primarily responsible for taking a screenshot of the webpage as rendered by the browser. In order to ensure that the resulting screen capture is identical to what the user of the browser sees, TrueClick copies every pixel displayed in Firefox's main browsing window in a hidden HTML *canvas* internal to the browser, and dumps the results into an image file.

This component is also tasked with providing the necessary information for the Feature Extractor to identify the visual difference in multiple views of the same webpage. To this end, once TrueClick is activated, the Screen Grabber issues an additional HTTP request for the displayed webpage, renders it in a hidden browser window, and uses this to take a second screenshot of the page with (potentially) different banners. Note that during this process, care must be taken to ensure that the dimensions and display properties of the hidden window are identical to that of the original window so that the two screenshots obtained match and the subsequent comparison can be carried out accurately.

6.3 Image Grabber

The Image Grabber component identifies the image files referenced by the webpage, and builds a temporary image database to be used by the Image Extractor to match fragmented banners against. However, simply parsing the DOM tree of the webpage is not an effective way of accomplishing this task since many banners are displayed dynamically after page load by JavaScript ad libraries. Similarly, even

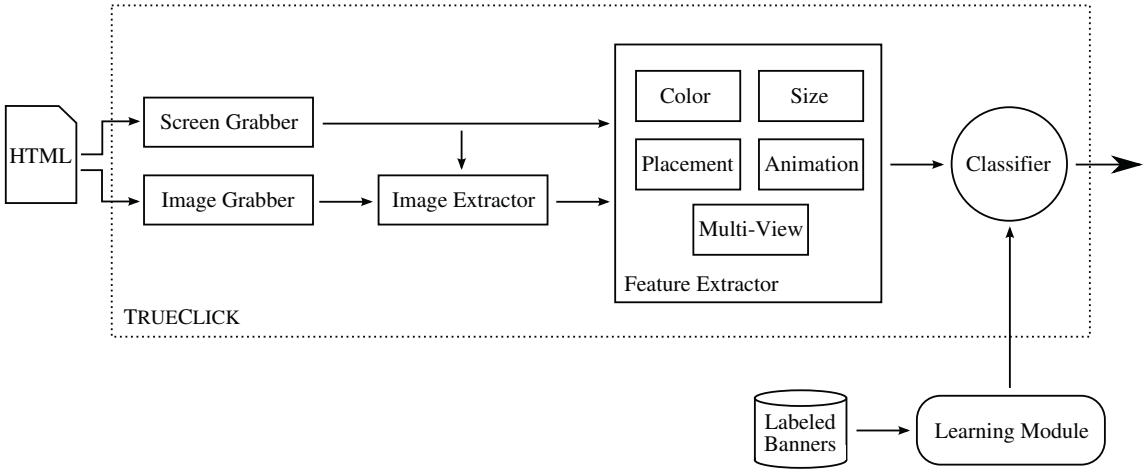

Figure 4: Overview of the architecture and various components of TrueClick.

when the URLs to the image files can be detected, downloading the banners from those locations is not a reliable way of obtaining the images, because the URLs provided by ad delivery frameworks often rotate between different banners and serve different image files with each request.

In order to address these problems, Image Grabber transparently intercepts HTTP responses from web servers and inspects the payload. Once it has been determined that the data corresponds to an image file, it is copied and inserted to the image database. This technique has the additional benefit of avoiding downloading the same image files a second time, saving bandwidth and allowing for quick detection of trick banners.

7. EVALUATION

In this section, we describe our experiments to measure the accuracy and usability of our solution in identifying trick banners in the wild. We evaluated the effectiveness of our classifier on a data set of banners we collected, conducted a user study to demonstrate that TrueClick is practical, and finally, compared the detection effectiveness to an existing ad detection system, AdBlock Plus.

7.1 Data Collection

To train and evaluate our system, we collected trick banner samples and images of genuine download buttons from popular file sharing websites, including one-click hosters, bittorrent sites, and online video streaming sites. Our training data set consists of only English websites, while the evaluation data set contains both English and non-English websites. We chose to perform the data collection and labeling procedure manually instead of crawling these websites, so that we could use cues from the pages to determine whether the collected samples are trick banners or not with high confidence and train our classifier with an accurate data set. We reached the actual file download pages by searching for popular movies and computer programs at the file sharing websites and services. In the banners we collected, we looked for keywords that could be intended to trick users such as *Download*, *Watch*, *Now*, *Save*, *Play*, *Get it*, and *Store*.

We note that such file sharing websites are not exclusively used for illegal media trading, but are also often used to distribute media to large audiences (e.g., software updates, non-commercial documentaries). Hence, our aim is to protect users in general, even though some might be engaged in illicit behavior when they are tricked by malicious banners.

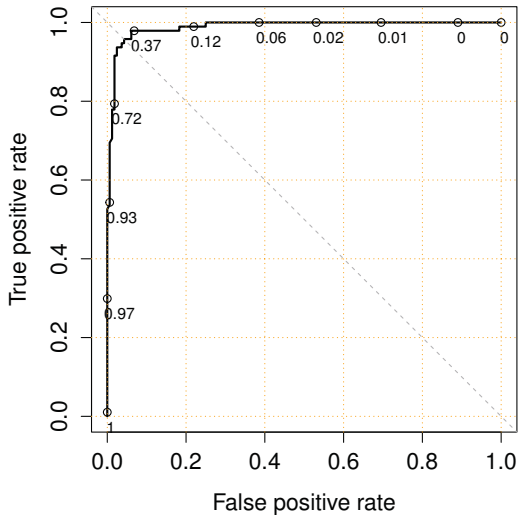

Figure 5: ROC curve for a 10-fold cross validation of TrueClick's random forest classifier.

To train and evaluate our classifier, we used 165 trick banners and 94 images corresponding to genuine download links for a total of 259 banner samples from 88 file sharing websites. For the comparison with AdBlock Plus, we used a disjoint set of 415 trick banners collected from 82 websites. In total, we collected 674 banner samples from 170 file sharing websites.

7.2 Evaluation of the Classifier

To build a classifier to distinguish between trick banners and genuine download links, we used the R statistical machine learning environment and, specifically, the ipred package, to train a random forest classifier from which the importance of predictors were assessed.

Figure 5 displays a ROC curve for a 10-fold cross validation of the resulting random forest classifier over our training

Feature	Importance (bits)
x-position	23.37
y-position	25.91
Size	100.06
Color	28.71
Animation	21.98
Multiview	52.14

Table 1: Information gain for each feature.

Experiment	Correct Clicks	Incorrect Clicks	Median Score	Avg. Score
Original page	29	91	0	0.725
w/ TrueClick	103	17	3	2.575

Table 2: Results of the TrueClick user study, showing the number of correct and incorrect clicks. The presence of incorrect clicks when using TrueClick is due to trick banners missed by our system.

set, using a cutoff value of 0.1. An n-fold cross validation partitions the entire data set into n equal-sized samples, or folds, trains on $n-1$ folds, and then validates the resulting model on the remaining fold. This process is repeated for each fold. The ROC curve plots the true positive rate against the false positive rate of the best-performing classifier as the discrimination threshold – i.e., the boundary between the trick banner and genuine link classes – is varied over $[0, 1]$. The value of the ROC curve lies in the guidance it provides in selecting thresholds to bias towards true positives at the expense of false positives, and vice versa.

The ROC curve shows that our classifier achieves a 96.97% true positive rate given a false positive rate of 3.03%, which is lower than the critical false discovery rate threshold of 5%. In other words, 3.03% of trick banners were incorrectly classified as correct banners. As shown in Table 1, the size feature is the most effective in distinguishing trick banners from legitimate download links. The table is generated using out-of-bag samples from the training data.

7.3 Effectiveness of Trick Banner Detection

We tested TrueClick's usability and effectiveness in guiding users to identify and click on the genuine download links on a file sharing website by conducting a user study, and comparing its detection performance to AdBlock Plus.

User Study. We performed our user study on 40 undergraduate and graduate computer science students. While we did not explicitly evaluate the participants' technical savviness, it is reasonable to expect them to be relatively expert computer and Internet users.

We first briefed all participants that they were going to take part in a user study on identifying the genuine download links on English-only file sharing websites, and then instructed them to click on the link or button they thought was legitimate on the webpages they were shown. Next, we presented each participant with unmodified pages from three different file sharing websites for them to perform this task on. In order to control for the fact that advertisement banners change every time a page is requested, we did not use the actual websites in our test, but instead created identical offline replicas with a fixed set of banners. Once the participants completed the first three tasks, we repeated the experiment using the same three websites, but this time in a browser window running TrueClick so that our system could analyze the page content and mark detected trick banners as such. We observed each participant complete all six tasks, recorded the number of correct and incorrect clicks for each of them, and assigned scores based on the number of correct clicks. The results are shown in Table 2.

These results demonstrate that in the experiments where the participants were assisted by TrueClick, there was a **3.55** factor improvement in the scores on average. We also checked these results for statistical significance using a standard paired difference test, namely, the Wilcoxon signed-rank test. The results of the test ($V = 0$, $p < 4.82 \times 10^{-7}$) confirm that the scores obtained with and without TrueClick indeed constitute non-identical populations.

Comparison to AdBlock Plus. Existing systems such as Adblock Plus are capable of identifying and blocking trick banners in some cases. To understand whether TrueClick is necessary given the existence of such systems, we compared its detection effectiveness with Adblock Plus.

We conducted experiments with 415 manually-identified trick banners from 82 websites that were not used in the training phase of the previous experiments. TrueClick correctly identified **380 (91.6%)** of these as trick banners using the previously-generated classifier without extra training or manual tuning, whereas Adblock detected 190 (45.8%). In contrast, there were only 8 banners detected by Adblock Plus, but not TrueClick.

We stress that these result do not suggest TrueClick is a substitute for Adblock Plus. TrueClick, as discussed and evaluated in this paper, focuses on detection of trick banners, and this experiment is not sufficient to draw conclusions on its ability to detect ordinary, benign advertisements. Therefore, we conclude that while Adblock Plus provides an efficient filter against general Internet advertising, supporting it with TrueClick significantly improves protection against potentially malicious trick banners.

8. DISCUSSION

The system we propose for automated trick banner detection has some limitations that we highlight here. First, we stress that in contrast to other work on trick banners and attacks against users, TrueClick is intended to address the specific case of images that masquerade as genuine download or play links on webpages. While the techniques we make use of here could be extended to cover other attacks, that is not the focus of this work.

As we noted in our evaluation of the random forest classifier TrueClick uses, the models we are able to build over our data set generate a non-trivial number of false positives. However, our models nevertheless classify the majority of trick banners correctly even when the threshold is set to produce a 0% false positive rate. Therefore, this represents a significant reduction in the number of trick banners that users must navigate, and our user experiments demonstrate that this translates to much better security decisions in practice when TrueClick is deployed.

We note that due to variations in content between consecutive page loads, we have observed our system to classify some *non-clickable* regions of the page as trick banners. However, due to the way in which we deploy our classifier

in the browser, this does not have generally have a deleterious effect on the user experience as the mislabeled content is non-interactive and obscuring it does not affect the functionality of the page.

Finally, an unlikely but interesting limitation of our system we discovered during our experiments involved a small number of image files that contained a corrupt GIF header. Although these files could successfully be displayed in a browser window, attempting to run our analysis on them caused the image processing library in our implementation to fail and abort the detection process. We only encountered three such images over the course of our experiments. After manual analysis of the files, we concluded that the images were likely created by a buggy image editor. Still, this observation demonstrates that purposely injecting errors inside image files could be used by trick banner creators as an evasion technique against automated analysis by TrueClick and similar tools, and highlights the importance of building an implementation with analysis routines robust against errors in image file headers.

9. CONCLUSION

In this work, we have highlighted the problem of trick banners that masquerade as benign links to download files or play videos by emulating the visual characteristics of the correct links. We have defined a number of visual features that distinguish trick banners from genuine links, including image size, color, placement on the enclosing webpage, whether they contain animation effects, and whether they consistently appear with the same visual properties on consecutive loads of the webpage. Using these features, we have built TrueClick, which uses image processing and machine learning to automatically detect trick banners. Our approach operates purely over visual features and, after an initial training, requires no further manual effort, for instance to compile blacklists as current approaches do.

We have evaluated our system over a data set of manually labeled trick banners and benign image links. Our experiments showed that our classifier achieves a 96.97% true positive rate given a false positive rate of 3.03%, showing that TrueClick can correctly detect the majority of the trick banners on file sharing websites with a reasonable low false positive rate. We tested our implementation of TrueClick on 40 users, and found that TrueClick resulted in a 3.55 factor improvement in correct link selection. We also demonstrate that TrueClick serves as an effective and useful complement to existing approaches for identifying trick banners such as Adblock Plus. We conclude that TrueClick successfully assists even technically-sophisticated users in correctly selecting benign image links despite the presence of malicious trick banners.

It remains an open question whether TrueClick could be supported with static analysis of webpages in order to further improve detection effectiveness, which is a promising direction for future research.

Acknowledgment

This work was supported by the Office of Naval Research (ONR) under grant N000141210165, and Secure Business Austria.

10. REFERENCES

[1] Adblock Plus. `https://adblockplus.org`.
[2] ADSafe. `http://www.adsafe.org`.
[3] Privoxy. `http://www.privoxy.org`.
[4] Squid. `http://www.squid-cache.org`.
[5] E. Adar, D. S. Tan, and J. Teevan. Benevolent Deception in Human Computer Interaction. In *CHI*, 2013.
[6] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.
[7] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.
[8] J. Chandon and M. S. Chtourou. Factors Affecting Click-Through Rate. In C. P. Haugtvedt, K. A. Machleit, and R. Yalch, editors, *Online Consumer Psychology: Understanding and Influencing Consumer Behavior in the Virtual World*, chapter 6. Psychology Press, Jan. 2005.
[9] T. Chen, S. Dick, and J. Miller. Detecting Visually Similar Web Pages: Application to Phishing Detection. *ACM TOIT*, 10(2):5:1–5:38, June 2010.
[10] G. Conti and E. Sobiesk. Malicious Interface Design: Exploiting the User. In *WWW*, 2010.
[11] M. Finifter, J. Weinberger, and A. Barth. Preventing Capability Leaks in Secure JavaScript Subsets. In *NDSS*, 2010.
[12] K. Gallagher and J. Parsons. A Framework for Targeting Banner Advertising on the Internet. In *HICSS*, 1997.
[13] F. Gargiulo and C. Sansone. Combining Visual and Textual Features for Filtering Spam Emails. In *ICPR*, 2008.
[14] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.
[15] Google. Google Caja. `https://developers.google.com/caja`.
[16] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe Exposure Analysis of Mobile In-App Advertisements. In *ACM WiSec*, 2012.
[17] C. Grier, S. Tang, and S. T. King. Secure Web Browsing with the OP Web Browser. In *IEEE S&P*, 2008.
[18] J. Grossman and M. Johansen. Million Browser Botnet. Black Hat USA, 2013.
[19] S. Guarnieri and B. Livshits. Gatekeeper: Mostly Static Enforcement of Security and Reliability Policies for Javascript Code. In *USENIX Security*, 2009.
[20] S. Guha, B. Cheng, and P. Francis. Privad: Practical Privacy in Online Advertising. In *USENIX NSDI*, 2011.
[21] H. Haddadi, P. Hui, and I. Brown. MobiAd: Private and Scalable Mobile Advertising. In *ACM MobiArch*, 2010.
[22] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer Science+Business Media, LLC, New York, NY, USA, 2nd edition, 2009.

[23] D. Hopwood. Jacaranda. http://www.jacaranda.org.

[24] L. Huang, A. Moshchuk, H. J. Wang, S. Schechter, and C. Jackson. Clickjacking: Attacks and defenses. In *USENIX Security*, 2012.

[25] InformAction. NoScript. http://noscript.net.

[26] Interactive Advertising Bureau. Ad Unit Guidelines. http://www.iab.net/standards/adunits.asp.

[27] Interactive Advertising Bureau. IAB Internet Advertising Revenue Report. 2012 Full Year Results. http://www.iab.net/media/file/IAB_Internet_Advertising_Revenue_Report_FY_2012_rev.pdf.

[28] A. Juels. Targeted Advertising ... And Privacy Too. In *Topics in Cryptology: CT-RSA*. Springer-Verlag, 2001.

[29] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo. Don't Kill My Ads!: Balancing Privacy in an Ad-supported Mobile Application Market. In *HotMobile*, 2012.

[30] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang. Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising. In *ACM CCS*, 2012.

[31] S. Maffeis and A. Taly. Language-Based Isolation of Untrusted JavaScript. In *IEEE CSF*, 2009.

[32] Marketing Terms. Trick banner definition. http://www.marketingterms.com/dictionary/trick_banner/.

[33] M. Maurer and D. Herzner. Using Visual Website Similarity for Phishing Detection and Reporting. In *CHI Extended Abstracts*, 2012.

[34] E. Medvet, E. Kirda, and C. Kruegel. Visual-Similarity-Based Phishing Detection. In *SecureComm*, 2008.

[35] Microsoft. Microsoft Web Sandbox. http://www.websandbox.org.

[36] M. Mimoso. Malware Campaign Leverages Ad Networks, Sends Victims to Blackhole. http://threatpost.com/malware-campaign-leverages-ad-networks-sends-victims-to-blackhole, Sept. 2013.

[37] K. Onarlioglu, U. O. Yilmaz, E. Kirda, and D. Balzarotti. Insights into User Behavior in Dealing with Internet Attacks. In *NDSS*, 2012.

[38] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner. AdDroid: Privilege Separation for Applications and Advertisers in Android. In *ASIACCS*, 2012.

[39] C. Reis and S. D. Gribble. Isolating Web Programs in Modern Browser Architectures. In *EuroSys*, 2009.

[40] A. Reznichenko, S. Guha, and P. Francis. Auctions in Do-Not-Track Compliant Internet Advertising. In *ACM CCS*, 2011.

[41] Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. 40(2):99–121, 2000.

[42] W. Salusky. Malvertising. http://isc.sans.edu/diary/Malvertising/3727, June 2007.

[43] U. Shankar and C. Karlof. Doppelganger: Better Browser Privacy without the Bother. In *ACM CCS*, 2006.

[44] S. Shekhar, M. Dietz, and D. S. Wallach. AdSplit: Separating Smartphone Advertising from Applications. In *USENIX Security*, 2012.

[45] Symantec. Malware Security Report: Protecting Your Business, Customers, and the Bottom Line. www.verisign.com/verisigntransition101/files/MalwareSecurityReport.pdf, 2011.

[46] M. Ter Louw, K. T. Ganesh, and V. N. Venkatakrishnan. AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements. In *USENIX Security*, 2010.

[47] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy Preserving Targeted Advertising. In *NDSS*, 2010.

[48] H. J. Wang, C. Grier, A. Moshchuk, S. T. King, P. Choudhury, and H. Venter. The Multi-Principal OS Construction of the Gazelle Web Browser. In *USENIX Security*, 2009.

[49] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng. Detection of Phishing Webpages Based on Visual Similarity. In *WWW*, 2005.

[50] L. Zeltser. Malvertising: Some Examples of Malicious Ad Campaigns. http://blog.zeltser.com/post/6247850496/malvertising-malicious-ad-campaigns, June 2011.